VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

PLÁNOVÁNÍ TRAS MULTIROBOTICKÉHO SYSTÉMU V DYNAMICKÉM PROSTŘEDÍ

MULTIROBOT PATH PLANNING IN A DYNAMIC SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **LADISLAV DOKOUPIL**
AUTHOR

VEDOUCÍ PRÁCE — **Doc. Ing. ZBOŘIL FRATIŠEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Dokoupil Ladislav**
Program: Informační technologie
Název: **Plánování tras multirobotického systému v dynamickém prostředí**
Multirobot Path Planning in a Dynamic System
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se se současnými metodami skupinového plánování cest.
2. Pro systém s více roboty-agenty pracující v dynamicky se měnícím prostředí zvolte nebo navrhněte takové, které povedou k racionálnímu koordinovanému chování agentů vzhledem k cílí zaujmout pozice do určitého času. Uvažujte možné konflikty mezi roboty a možnost nutnosti přehodnocování naplánovaných tras.
3. Implementujte zvolené metody a ověřte jejich fungování ve vhodně zvoleném systému, nejlépe v systému soutěže MAPC z let 2019 a 2020.
4. Porovnejte fungování zvoleným metod pro různé parametry dynamiky prostředí, případně oproti již existujícím řešením, pokud existují. Zhodnoťte dosažené výsledky a diskutujte možná vylepšení do budoucna.

Literatura:

- Wooldridge, M.:An Introduction to MultiAgent Systems, Wiley, 2009

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Zbořil František, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 29. července 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Tato práce řeší problematiku prohledávání dynamického prostředí s využitím multiagentních systémů.

Primárním výsledkem této práce je zapojení do soutěže MAPC2022, ale uplatnění lze nalézt v prohledávání neznámého prostoru za předpokladu omezené viditelnosti a zároveň neomezené vzdálenosti komunikace agentů. Po popsání současných možností řešení dané problematiky i s jejich omezeními je k implementaci zvolen algoritmus na bázi optimalizace mravenčí kolonie.

Z údajů sbíraných při běhu programu s různými parametry byly následně vytvořeny přehledné grafy. Výsledkem práce je optimalizace platformy předchozího roku, lepší synchronizace agentů a až o polovinu lepší výsledky z pohledu množství prozkoumaného terénu oproti předchozímu řešení.

Abstract

This thesis deals with the problem of dynamic environment search using multi-agent systems.

The primary result of this work is participation in the MAPC2022 contest, but the application can be found in the exploration of unknown space, assuming finite visibility and unlimited distance of communication of the agents. After describing the current methods for solving the given problem and their limitations, an algorithm based on ant colony optimization is proposed.

Graphs were then created with data from running program with various parameters. The result of work is agents synchronization improvements and overall optimization of the platform involved in the mentioned contest from previous year. As a result half more of explored space was measured compared to previous solution.

Klíčová slova

Multiagentní systém, Průzkum, Koordinace robotů, Agenti, Komunikace, Plánování cest, Dynamický systém

Keywords

Multi-agent system, Exploration, Multi-robot cordination, Agents, Communication, Path-planning, Dynamic system

Citace

DOKOUPIL, Ladislav. *Plánování tras multirobotického systému v dynamickém prostředí.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Zbořil Fratišek, Ph.D.

Plánování tras multirobotického systému v dynamickém prostředí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Fratiška Zbořila Ph.D.. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Ladislav Dokoupil
24. července 2022

Poděkování

Tímto bych chtěl poděkovat Doc. Ing. Fratišku Zbořilovi Ph.D. za vedení práce a odborné konzultace při průzkumu řešení a jejich implementaci. Dále bych rád poděkoval rodině, která mě po celou dobu studia jak finančně tak psychicky podporovala.

Obsah

Kapitola 1

Úvod

Průzkum terénu je problém starý jako lidstvo samo. Již odedávna bylo třeba lokalizovat užitečné objekty okolního světa k zajištění přežití. Pro rychlejší průzkum je vhodné nasazení více průzkumníku, což ale s sebou přináší problém jejich organizace.

V současné době se stále více zkoumají distribuované systémy, jež umožňují řešit problémy, které by pro jednotlivé agenty byly složité či přímo neřešitelné, nesou ale s sebou i problémy jako spolehlivou vzájemnou komunikaci, kooperaci a splnění cílů, které se obecně mezi agenty mohou lišit. V této práci je řešena problematika spolupráce agentů při kontinuálním prohledávaní dynamického prostředí.

Práce je inspirována mezinárodní soutěží multiagentních systémů MAPC, neboli The Multi-Agent Programming Contest. Soutěž je každoročně pořádána již od roku 2005. Soutěže se účastní řada univerzit z celého světa včetně VUT FIT od roku 2018. Týmy zde podle předem určeného scénáře navrhují multiagentní systém. Cílem soutěže je plnit úkoly, které jsou sdíleny mezi týmy a pouze nejrychlejší z nich získá body. Letos je cílem agentů seskupovat bloky do zadaných tvarů a jejich přemístění na cílové pozice. Agenti navíc mají omezenou viditelnost a neznají svou globální pozici, proto si musí vybudovat mapu prostředí, ve kterém se navíc dynamicky mohou měnit překážky.

V následující sekci 2 je nejprve formálně vysvětlen pojem multiagentní systém. Součástí tohoto systému je obecně různorodé prostředí, jehož charakteristiky a odlišnosti jsou v části 2.1 nastíněny. Načež je definován pojem agenta s jeho základními vlastnostmi dle [17] a komunikační prostředky, které tito agenti využívají. Sekce 3 zkoumá současné řešení problematiky průzkumu prostředí multiagentními systémy s různými specifiky. Těmito odlišnostmi je zejména myšlena rozdílnost práce ve statickém a dynamickém prostředí, ale také je kladen důraz na odlišné charakteristiky komunikace a různé vnímání prostředí jednotlivými agenty. Sekce 4 pojednává o současném stavu platformy FIT z předchozího ročníku soutěže a návrhu algoritmu, implementaci programu do zmíněné platformy a o problémech s tím spojených. V kapitole 5 je testována funkčnost a chování algoritmu. Toto testování bylo prováděno sběrem dat z běhů programu s různými parametry pro jejichž vyhodnocení a grafické zobrazení byl následně vytvořen python skript. Celkové shrnutí výsledku práce se nachází v sekci 6.

Kapitola 2

Multiagentní systémy

Informace v této sekci jsou převzaty z [13], [17] a [9]. První práce zmiňující multiagentní systémy se datují do 80. let minulého století a od té doby zájem o ně pouze roste. Dle [13] je tento zájem způsoben zejména přesvědčením, že agenti jsou vhodným softwarovým paradigmatem k práci a využití masivních distribuovaných systémů jako je třeba internet, které se v této době začaly rozvíjet a umožňovat lidstvu přistup k většímu výpočetnímu výkonu. Dále by se tento zájem dal připsat i pozornosti, která se dostává oboru umělé inteligence, jelikož multiagentní systémy by se daly považovat za její distribuovanou formu.

Původ myšlenky agentních systémů jako návrhu založeném na dekompozici problému na autonomní jednotky by šel jistě nalézt v již zmíněné distribuované umělé inteligenci, ale klidně i v základní myšlence OOP. S trochou fantazie by šel nalézt i v přírodě, kde lidé či obecně zvířata jsou nucena kooperovat k dosažení vyšších cílů, pro jednotlivce nedosažitelných. Aby tato myšlenka byla prakticky realizovatelná, je nejdříve třeba daný úkol dekomponovat na autonomní jednotky, což ale nemusí být vždy možné. Odměnou za tuto snahu je jednodušší popis a návrh jednotlivých komponent. Zároveň nám takový návrh umožnuje řešit komplexnější úlohy, ať už z důvodu obtížnosti návrhu monolitického systému, či kvůli nedostatku výpočetního výkonu samotného procesoru. Přidání více prvků do systému (horizontální škálování) je totiž obecně jednodušší než zvýšení výkonu jednoho prvku (vertikální škálování). Problémem návrhu je taktéž nepředvídatelnost systému, jelikož agenti vzájemně komunikují a mohou mít obecně různé cíle a systém je z pohledu pozorovatele méně transparentní než obdobná soustava řízená centrálním prvkem.

Pojem multiagentního systému kvůli svému širokému uplatnění nemá jednoznačnou definici, ale pro naše účely jej lze uvést dle [13] jako systém sestavený z množiny agentů, kteří spolu interagují pomocí vzájemného propojení, např. počítačovou sítí. Agent je zde většinou počítačový systém konající pod záminkou uživatele. Jednotliví agenti systému obecně nemusí být homogenní a mohou mít různé motivace a cíle. Každý z agentů pak musí být schopen alespoň těchto dvou činností. První z nich – *autonomie* je schopnost agenta rozhodovat o svých činnostech nutných k dosažení cíle a schopnost konat samostatně, místo toho, aby mu bylo nutno v každém okamžiku explicitně přiřadit činnost. *Interakce* je druhá nutná činnost agenta a rozumíme jí nejen vzájemnou výměnu dat, ale i další sociální chování jako spolupráci, koordinaci a řešení konfliktů.

Multiagentní systém obdobně jako spousta dalších odvětví umělé inteligence je relativně nový a neustále se vyvíjející obor počítačových technologií. Stále se pro ně hledají nová uplatnění, příkladem praktického využití může být distribuované sledování vozidel [8]. Agenti zde reprezentují senzory s omezeným výhledem, které byly rozptýleny po prostoru,

byla tedy nutná komunikace senzorů pro analýzu pohybu vozidla celým prostorem. Podobné systémy se také používají na sledování pohybu letadel.

2.1 Prostředí

Prostředí agentních systémů lze chápat jako systém $S =< U, R >$, tedy množinu prvků U_{env} a množinu relací R_{env} mezi nimi. Důležitým aspektem při návrhu systémů je prostředí, v němž je agent situován a specifika s daným prostředím spojená.

2.1.1 Spojité nebo nespojité

Dle časové množiny, se kterou agent pracuje, lze prostředí rozdělit na diskrétní a spojité. Je-li systém modelován se spojitým časovým intervalem, mluvíme o spojitém prostředí. Modelujeme-li systém pouze pod určitými časovými kroky, hovoříme o diskrétním prostředí.

2.1.2 Dynamické nebo statické

V případě, že všechny změny systému jsou zapříčiněny pouze akcemi agentů, jedná se o prostředí statické. Prostředí je dynamické, pokud jsou některé změny prostředí nezávislé na akcích agentů, jako v případě letošního ročníku soutěže MAPC.

2.1.3 Deterministické nebo nedeterministické

V případě opakovaných simulací se stejnou počáteční konfigurací musí v deterministickém prostředí mít simulace vždy stejný průběh a koncový stav. V opačném případě se jedná o prostředí nedeterministické.

2.2 Agent

Informace v této sekci jsou převzaty z [13] a [17]. Pojem agent nemá jednoznačnou definici, díky rozsahu domén, ve kterých je použit. Pro snadnější pochopení můžeme pro naše účely použít definici z [14] “Agent je počítačový systém, jež je situován v nějakém prostředí a je schopen autonomní činnosti v daném prostředí za účelem splnění svých úkolů”. Ačkoliv agent může obecně nabývat různých forem, od lidí přes roboty až po počítačový software, zmíněné principy a algoritmy zde uvedeny jsou nezávislé na jeho formě. Hierarchii schopností agentů můžeme dle Wooldridge a Jenningse klasifikovat pomocí rozvinutosti následujících vlastností:

- *autonomie* je schopnost samostatného a nezávislého jednání na základě svého rozhodnutí, agent se navíc této schopnosti může vzdát při kooperaci s dalšími agenty
- *reaktivita* je schopnost vnímat své okolí a reagovat na jeho změny
- *proaktivita* je schopnost si stanovovat cíle a dle nich adekvátně jednat
- *sociální schopnosti* agent zde nejedná pouze sám za sebe, ale ke splnění svých či společných cílů je schopen komunikovat a spolupracovat s dalšími agenty

Chování agentů

Dle [9] lze architekturu agenta rozdělit na 4 moduly:

- modul senzorů - vstupní rozhraní agenta. Senzory jsou jediným způsobem, kterým agent přijímá vjemy z okolního prostředí, což zahrnuje i příjem zpráv od ostatních agentů
- modul vnitřní reprezentace - paměť agenta, vjemy jsou zde zpracovány do vnitřní reprezentace
- modul plánování - řídí chování agenta, obecně je dle tohoto modulu určena inteligence agenta
- modul aktuátoru - výstupní rozhraní agenta pomocí nějž ovlivňuje agent systém

Dle práce s výše zmíněnými moduly lze agenty rozdělit na následující typy:

- reaktivní agenti - za nejjednodušší by se dal považovat agent reaktivní, který neuchovává stav prostředí, provádí tedy pouze předem určené akce pouze na základě svých vjemů
- kognitivní agenti - dají se považovat za protiklad reaktivního agenta, z vypozorovaného prostředí si vytváří vlastní bázi dat. Z ní je schopen vyvozovat závěry, učit se a odvozovat své další akce
- deliberativní agenti - mají schopnost plánování a výběru z dlouhodobějších cílů. K dosažení daného cíle tedy musí být schopni s využitím vnitřní logiky vyvodit a vykonat nutnou posloupnost akcí. K dosažení tohoto cíle je schopen ovlivňovat prostředí, tato vlastnost je nazývaná proaktivita
- racionální agenti - kombinují schopnosti všech agentů předcházejících, tedy udržují si stav prostředí, plánují a zároveň jsou schopni vyvozovat závěry

2.3 Komunikace a řešení konfliktů

Informace v následující sekci pochází zejména z prací [16], [9], [6] a [7]. Aby agenti byli schopni spolupracovat, musí být schopni komunikovat. K tomu potřebují komunikační jazyk, popis struktury zpráv a stejné vnímání prostředí. Takováto komunikace může být dvojího druhu, *nepřímá* a *přímá*.

Nepřímá funguje pomocí ovlivňování stavu prostoru a tedy vjemů dalších agentů.

Přímá je komunikace mezi agenty, k jejímž základním účelům patří:

- *Dotazování* – snaha o získání informace od agenta, kterému věří, že danou informaci může poskytnout
- *Hledání informace* – společné pátrání po znalosti, která není známa žádnému z agentů
- *Přesvědčování* – snaha přesvědčit agenta k přijetí znalostí jiného agenta
- *Vyjednávání* – debata o podmínkách sdílení prostředků, či poskytnutí služeb k dosažení maximálního zisku všech zúčastněných

- *Porada* – snaha nalézt řešení problému zúčastněných. Agenti zde poskytují své znalosti, cíle a schopnosti za cílem dohodnutí se na optimalizaci dalšího postupu
- *Eristický dialog* – forma rozhovoru, během kterého si agenti vyměňují informace bez ohledu na jejich pravdivost čistě za účelem vítězství. Typickým příkladem takového dialogu je hádka

Pro usnadnění implementace komunikace byly vytvořeny různé jazyky a frameworky, ale jazyk Knowledge Query and Manipulation Language, dále označován jen jako KQML, by se dal považovat za předchůdce, z jehož myšlenek novodobé jazyky vychází. Samotná architektura použitá k předání zpráv stejně jako její kódování není přímo jazykem specifikována a může být provedena např. pomocí vzdáleného voláni procedur nebo pomocí webových služeb. Jazyk se zabývá především formátem zprávy se snahou o co nejobecnější využití. Veškerá komunikace probíhá pouze vůči virtuální bázi znalostí, jehož existence se předpokládá u každého z agentů, a značně tedy zjednodušuje implementaci oproštěním se od složitějšího zpracování zpráv. Důležitým pilířem těchto zpráv jsou předdefinované "performativy", které určují význam celé zprávy, příkladem může být zpráva:

```
(PERFORMATIVE
    :sender "agent A"
    :reciever "agent B"
    :content "temperature(30)"
    :language "Java"
)
```

Zpráva nedefinuje význam symbolu "temperature" ani jeho argumentů, znalost těchto významů musí mít agenti společnou. Tato zpráva může s využitím různých "PERFORMATIVE" nabývat odlišných významů, například pro výše zmíněnou zprávu:

- *ask-if* se dotazuje agenta, zda je teplota 30° C
- *insert* vkládá agentovi B do báze znalostí, že teplota činí 30° C, naproti tomu v případě *tell* se pouze oznamuje a agent se znalostí zachází dle vlastní iniciativy
- *achieve* žádá agenta, aby učinil kroky k dosažení dané teploty

KQML komunikace díky svému asynchronnímu charakteru umožnuje vytvořit decentralizovaný systém, naopak v případě klasického OOP volání je udržována synchronizace zejména kvůli návratovým hodnotám. Výhodou tohoto přístupu je možnost komunikace "on demand", která nezatěžuje agenta nepotřebnou komunikací, ale pouze takovou, kterou si vyžádal. Například zpráva uvedená výše je jednoduchého charakteru a dala by se snadno nahradit systémovým voláním, ale zprávy broadcastového charakteru zde lze nahradit pomocí performativů *advertise* a *subscribe* podobných TCP/IP multicastu, které umožnují vyhlásit přenos a přihlášení k příjmu streamu dat.

Při masivně distribuované komunikaci se stává výše zmíněný broadcast, tedy zasílání dat všem neefektivní a nabízí se hledat vhodnější způsoby navázání komunikace. Pasivním prostředkem umožňujícím agentům sdílet nově získané znalosti, žádosti pro vykonání služby, či nabízené služby je *nástěnka*, na kterou agenti informace umisťují a v případě potřeby i hledají. Například pro průzkum terénu by agent nejdříve zkoumal, zda nástěnka již neobsahuje informace o daném prostoru. Při neúspěchu by zkoumal, zda není v blízkosti jiný agent poskytující službu průzkumu, nebo by vystavil požadavek na průzkum dané oblasti.

Častěji používanou možností je *komunikace pomocí prostředníka*. Prostředník je agent či skupina agentů, kteří shromažďují schopnosti, záměry a přání jednotlivých agentů. Tyto znalosti si prostředník uchovává ve své bázi dat a díky nim je schopen zprostředkovávat požadavky. V případě KQML jazyka jsou tomuto typu komunikace vyhrazeny performativy:

- *broker-one/broker-all* - prostředník předá požadavek vhodnému agentovi a následně jeho výsledek předá žadateli.
- *recommend-one/recommend-all* - pracuje obdobně jako broker, ale odpovídající agent komunikuje přímo se zadavatelem dotazu a odlehčuje tedy práci prostředníka.
- *recruit-one/recruit-all* - prostředník hledá agenta, který je schopen na požadavek odpovědět, následně jej předá žadateli, který poté službu zpracuje bez pomoci prostředníka.

Je důležité, aby prostředník měl přehled o kompletním seznamu agentů v systému, jelikož veškerá komunikace probíhá přes něj, což s sebou nese i nevýhodu zhroucení komunikačního systému v případě výpadku těchto agentů. Proto je v praxi často využíván tzv. *sociální model* kombinující výše uvedené přístupy. Agenti zde tvoří skupiny dle svých schopností, či jiných vhodných náležitostí. V rámci těchto skupin komunikují agenti pomocí broadcastu a až v případě, že žádný člen skupiny není schopen požadavek zpracovat je směrován na prostředníka dané skupiny. Tento prostředník následně sdílí žádost známým skupinám a poté je žadatel informován o výsledku z cizích skupin.

Po navázání komunikace potřebují agenti jednotný komunikační protokol k vhodnému rozdělení zdrojů, cílů a volbě strategií, jako je například:

Aukce

Je komunikační protokol využíván převážně v multiagentních sítích ke sdílení úkolů. Agenti zde mají na počátku aukce přidělenou počáteční sumu, se kterou obchodují. Zbožím aukce jsou zde úkoly jednotlivých agentů. Obvykle se pracuje s *anglickým* systémem dražby, kdy prodávající zvolí počáteční cenu dle svého ohodnocení úkolu, následně ostatní agenti přihazují dle svých ocenění načež vyhraje nejvyšší nabídka. Rychlejší alternativou je dražba *holandská*, kde prodávající zvolí nadhodnocenou částku, která se v průběhu kola dražby snižuje a vyhrává agent, který jako první částku přijme, za co možná nejnižší cenu.

Hlasování

Je využíváno jako protokol pro výběr z neprázdného množství strategií. Takovéto hlasování probíhá ve dvou kolech. Nejdříve nastává část shromažďování návrhů, kdy je navíc zvolen elektor pro řízení druhé časti hlasování. Ve druhé části je elektor každým agentem informován o své volbě, načež sečte hlasy a informuje volící agenty o zvoleném návrhu s nejvyšším počtem hlasů, dle kterého se budou následně do dalších voleb řídit.

Argumentace

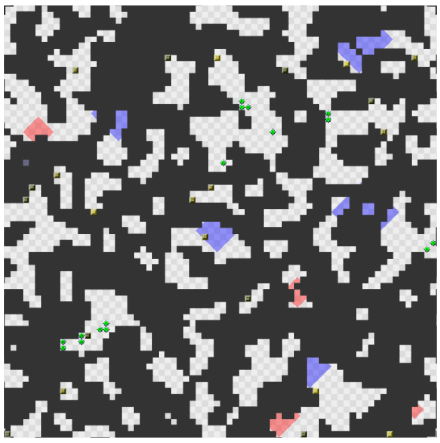
V případě, že agenti mají protichůdné informace využívá se argumentačního interakčního protokolu. Agenti se snaží názor podpořit/vyvrátit přesvědčením druhého agenta s využitím svých znalostí. Pokud se protistranu podaří přesvědčit akceptuje informaci a uloží ji do své báze dat.

2.4 2022: Agents Assemble III

The Multi-Agent Programming Contest [12] je soutěž pořádána každoročně již od roku 2005. Soutěž si klade za cíl stimulovat vývoj multiagentních systémů, toho má být docíleno soutěží v konkurenčním prostředí, kde spolu týmy musí soupeřit o splnění úkolu se sdílenými zdroji. Toto prostředí je popsáno v tzv. scénářích a jeho parametry jsou zasílány na počátku kola serverem, tudíž se může v průběhu soutěže měnit. Této soutěže se účastní řada univerzit z celého světa včetně od roku 2018 i tým VUT FIT.

Prostředí soutěže

Hlavními prvky letošní soutěže jsou agenti a bloky situované ve dvourozměrné síti buněk, kde každý agent a každý blok se nacházejí na právě jedné z těchto buněk a zároveň každá buňka dané sítě může být obsazena maximálně jednou entitou viz. 2.1. Mezi tyto entity patří: agent, blok, vydavač, překážky nebo volný prostor. Agenti soupeřících týmů také pro větší férovost začínají na stejné pozici, to je jediná instance, kdy může buňku mapy obývat více než jedna entita. Prostředí je navíc *předem neznámé velikosti*, jež se do sebe horizontálně i vertikálně uzavírá (tzn. agent se při přechodu přes levý "roh" objeví na mapě vpravo). Pro agenta tedy působí nekonečně a nemá možnost si zjistit absolutní souřadnice v rámci mapy. Server také disponuje možnosti vytvářet po mapě náhodné clear akce vedoucí ke změně prostředí. Tyto akce ovlivňují jak odstraňování existujících překážek tak vytváření nových.



Obrázek 2.1: ukázka prostředí soutěže MAPC

Mimo výše zmíněné objekty se na mapě objevuje i koncept zón, jakožto oblastí se speciálními efekty kruhového tvaru s určeným průměrem. Mezi tyto zóny patří task board, zóna pro odevzdání úkolů (dále jen goal zone) a zóna pro změnu role (role zone), které budou popsány dále. Task board byl využit k přijímání úkolu agentem a následně jej ten stejný agent musel odevzdat, avšak pro letošní ročník byl tento typ zóny zrušen.

V soutěži se pracuje s Manhattanskou vzdáleností, tudíž má každá buňka právě 4 přímé sousedy, vůči kterým lze vykonávat akce. Pokud se ve vzdálenosti jedna od agenta nachází blok, může si jej agent připojit na přilehlou stranu, takto si může tedy sám připojit maximálně čtyři bloky. Dva agenti nesoucí bloky se mohou spojit, pokud jsou tyto bloky v bezprostřední blízkosti a následně se může jeden agent z vzniklé struktury odpojit, a tak zanechá zbylého agenta s dvěma bloky. Tento proces může být opakován pro tvorbu

složitějších struktur. V průběhu simulace jsou náhodně generovány úkoly, které specifikují vyžadované tvary struktur bloků k doručení, časový limit a příslušnou odměnu. Tyto úkoly jsou sdíleny mezi soupeřícími týmy a odevzdat jej může pouze první, výše odměny se navíc snižuje se stářím úkolu. Soutěžící tým s nejvyšším skóre na konci běhu simulace vyhrává.

Agent soutěže

Prostředky a schopnosti, které má agent k dispozici, jsou dány rolí, která je mu přidělena. V rámci letošní soutěže mají agenti na výběr z *rolí*: default, worker, constructor, explorer a digger. Tyto role určují agentův dohled r, mobilitu *speed*, tzn. počet buněk, které je v jednom kroku simulace schopen překročit. Mobilita je závislá také na počtu nesených bloků, kdy je možné, že při velkém počtu se agent nebude moci pohnout. Agenti začínají s "default" rolí, která se vyznačuje $r = 5$, $speed = 1$ a $clear\_chance = 0.3$, jejíž význam bude popsán dále.

Každý agent nezávisle na roli začíná se stejnou energii, značenou celým číslem a s každým krokem simulace se jí část dobíjí. Energie je agentem využívána k provádění akcí a snižována je při zásahu protivníkem, clear akcí serveru nebo porušením pravidel agentem. Pokud tato metrika dosáhne nuly je agent dočasně deaktivován a odpojí se všechny připojené bloky.

V každém kroku může agent vykonat některou z následujících činností: skip, move, attach block, detach block, rotate, request to dispenser, submit task, clear, adapt role a survey. Za zmínku stojí akce clear, která je nejen využívána k čištění prostředí od překážek, ale také k útoku na cizí agenty, jelikož v případě úspěšného zásahu je agent na několik tahů deaktivován a jsou mu odpojeny všechny bloky. Survey poskytuje dodatečné informace o zkoumané buňce, či agentovi, který se na ní nachází a lze tedy použít k synchronizaci agentů do skupin.

Specifikace problému

Hlavní výzvou je tedy koordinace agentů k plnění úkolů. Počet nasazených agentů týmů se navíc s každým kolem mění, což může mít za následek změny strategie týmů. Agenti musí zjistit pozice a přemístit nutné bloky, kooperovat ke složení útvaru a rozdělit si mezi sebou nutné podúkoly. Bloky jsou získávány ze speciálních podavačů, které jsou náhodně bez znalosti agenta rozmístěny po mapě a vydávají vždy pouze jeden specifický typ bloku. Ke splnění úkolu může být nutné nejprve příslušné typy podavačů nalézt.

Dalším problémem je agentovo vnímání prostředí a jeho dohled, ten je omezen celým číslem r, které určuje Manhattanskou vzdálenost, po kterou nejdál agent dohlédne. Vjemy jsou agentům zasílány serverem na počátku každého kroku simulace ve formě pouze rozdílných vjemů od předchozích kol pro minimalizaci redundance internetové komunikace. V případě prvního kroku simulace či znovu připojení agenta jsou informace zaslány kompletně. Agenti jsou na počátku simulací náhodně rozmístěni po mapě, bez znalosti svých absolutních souřadnic a pozice svých kolegů, navíc nevnímají příslušnost agenta. Proto pokud se v dohledu potkají dva agenti, musí se nejdříve vzájemně identifikovat, aby mohli spojit své relativní znalosti vázané ke své počáteční pozici ve společnou bázi znalostí. Poté spolu mohou komunikovat na neomezenou vzdálenost.

Prostor mapy může obsahovat překážky bránící v průchodu. Tyto překážky jsou vygenerovány na počátku simulace, ale mohou se i dynamicky měnit. Agent se tedy může ocitnout obklopen překážkami, ze kterých se musí prokopat pomocí clear akcí, nebo vyčkat, zda pře-

kážka v budoucnu nezmizí. Tato dynamika prostředí pro agenty přináší nejistotu správnosti dlouho neprozkoumaných oblastí a tedy nutnost jejich opětovného průzkumu.

Pro větší dynamiku soutěže byl letos zaveden koncept norem. Normy slouží jako dočasné změny pravidel, jejichž porušení se trestá srážkou energie agentů. Omezení norem jsou stanovena na počet nesených objektů agentem nebo omezením maximálního počtu agentů se zvolenou rolí v rámci týmu.

Komunikace soutěže

Soutěžní prostředí vychází z architektury klient-server. Pořadatel soutěže umožnuje přístup ke kódu serveru a nabízí knihovnu v jazyce Java, která poskytuje jednotné rozhraní ke komunikaci se serverem. Na počátku kola se týmy autentizují na server a následně probíhá internetová komunikace v specifikovaném *json* formátu. Server zde agentovi zasílá vjemy, úkoly, normy a další dodatečné údaje, načež klient zasílá odpověď ve formě vyžádaných akcí agentů. Na tyto akce agentů server klienta vždy informuje o úspěchu či neúspěchu dané akce. Takováto výměna vjemů a akcí probíhá s každým krokem simulace až do konce kola, kdy je vyhlášen vítěz dle bodového zisku týmů.

Kapitola 3

Průzkum terénu

V Soutěži MAPC nejsou na počátku kola simulace známy žádné informace, ani velikost mapy, navíc je zde třeba pracovat s omezeným dohledem agenta. Pro hlídkování terénu by tedy bylo nejdříve vhodné zjistit velikost mapy, což obecně nemusí být vždy možné, či praktické tak musíme počítat s neznámou velikostí prostředí. Navíc kvůli dynamice terénu je třeba také udržovat aktuální informace, tedy hlídkovat již prozkoumaný terén, přičemž optimální strategie *objevování* a *hlídkování* se mohou lišit [5]. Výhodou soutěže je možnost komunikace agentů bez omezení vzdálenosti, aby však tento způsob komunikace mohl být využít musejí se agenti nejdříve synchronizovat. Takováto synchronizace probíhá pomocí společných vjemů, jelikož server nezasílá přímo identifikátory agentů v dohledu, pouze informaci o příslušnosti agenta do cizího či vlastního týmu.

3.1 Současné řešení

V současnosti je problematika objevování a hlídkování terénu multiagentním systémem otevřeným výzkumným tématem. K němu existuje celá řada odborných publikací. Kvůli různým předpokladům o prostředí (statické/dynamické nebo diskrétní/spojité) a agentech, kteří se liší zejména senzory a jejich dohledem, nebo možnostmi vzájemné komunikace, nejsou některá ze současných řešení k MAPC soutěži vhodná, přesto lze díky podobnostem na tato řešení nahlédnout a případně se inspirovat.

3.1.1 Mravenčí kolonie

Práce [3] se zabývá tvorbou algoritmu, který využívá agenty kombinující deliberativní a reaktivní chování pro průzkum a strážení terénu. Algoritmus vychází z třídy genetických algoritmů optimalizace mravenčí kolonie. Vzhledem k povaze agentů mravenců lze algoritmus využít k průzkumu jak statického, tak dynamického prostředí. Daná třída algoritmu obecně napodobuje chování mravenců při hledání potravy. Mravenci při průzkumu vypouští feromony a zároveň se snaží preferovat cesty, kde této látky cítí nejvíce a zřejmě zde tudíž prošlo nejvíce mravenců. Na pohled složité úkoly jsou tedy vykonány pomocí interakce a plnění jednoduchých úkolů skupinou mravenců. Pro správnou funkčnost daného postupu na rozdíl od reálných mravenců agenti preferují místa s minimálním obsahem feromonu, jelikož je žádoucí prozkoumání nových míst a ne nalezení optimální cesty k cíli. Při modelování jsou zohledňovány dvě skutečnosti feromonu. První je *difuze*, zajišťující šíření části pachové stopy do přímého okolí, aby ji mravenci mohli zachytit. Druhou zohledněnou skutečností

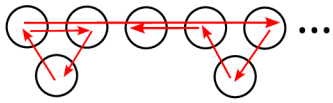
je *odpařování* feromonu, tedy s časem se množství pachové stopy snižuje a mravenci tudíž tuto cestu zvolí s nižší pravděpodobností.

Vypařování feromonů

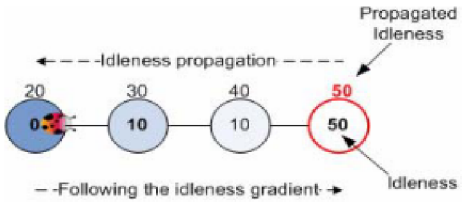
Každá buňka q je ohodnocena množstvím feromonu, který agent vypustí při návštěvě a s každým krokem simulace část p ubývá, dle předem daného vzorce:

$$q_{n+1} = q_n * (1 - p)$$

Tento proces zapříčiní, že nejdéle nenavštívená místa budou mít nejnižší ohodnocení. Agent tohoto algoritmu pracuje pouze s vizí omezenou na sousední pole, a tudíž při výběru cesty nahlédne na 4 sousední buňky a vybere tu s nejnižším feromonovým ohodnocením, pokud má více buněk stejné ohodnocení, vybere mezi nimi agent náhodně. Aby trasa pohybu nebyla příliš chaotická, zachová agent s vysokou pravděpodobností směr pohybu. Autor se dále zaobírá problematikou neoptimálností naivního přístupu tohoto algoritmu při určitých topologiích map. Problém může nastat v rozdělení cest, z nichž některá vede na již probádané území. jelikož se agent po průchodu zpětným cyklem raději vrátí k již starším probádaným územím, na místo průzkumu ostatních neprobádaných odboček grafu. Problém plyne z naivnosti algoritmu, jež ve výše zmíněné formě vybírá pouze z lokálních sousedních míst.



Obrázek 3.1: Ukázka problematiky cesty agenta při existenci cyklů grafu



Obrázek 3.2: Ukázka šíření pachové stopy na sousední buňky. Převzato z [3].

Difuze feromonů

Jako řešení předchozího problému cyklů se nabízí následující algoritmus, který navíc využívá propagace informace buňky od sousedů. Algoritmus využívá metriky nazvané "idleness", která označuje dobu od poslední návštěvy a její difuze sousedů. V šíření je zohledněna vždy vlastní buňka a přímý soused s nejvyšší hodnotou, formálně to lze vyjádřit:

$$OP_i = max(O_i, max(f(i,j)))$$

Kde O značí vlastní množství feromonu, OP její propagované množství a $f(i,j)$ je propagační funkce buněk j sousedících s i .

$$\text{If } OP_j - \alpha - \beta I(j) \geq OP_{min} \text{ Then} \quad f(i,j) = OP_j - \alpha - \beta I(j) \tag{3.1}$$

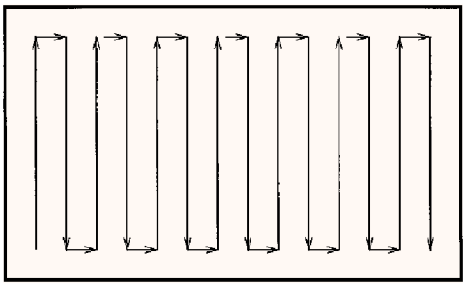
$$\text{Else If } OP_j > OP_{min} \text{ Then} \quad f(i,j) = OP_{min} \tag{3.2}$$

$$\text{Else} \quad f(i,j) = OP_j - 1 \tag{3.3}$$

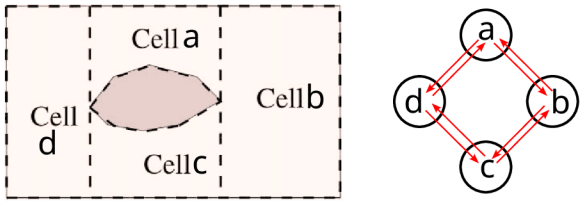
α značí koeficient propagace určující dosah šíření. Funkce $I()$ a koeficient β slouží k ukončení propagace, při naražení na agenta, aby nebylo zbytečně více agentů lákáno na stejné místo. Pomocí dodatečných podmínek zaručujeme, aby byl gradient feromonu klesající se vzdáleností od dané buňky. Pomocí této metriky se nám daří více zohledňovat vliv širšího okolí na průzkum agenta, místo pouhého přihlížení na sousední buňky.

3.1.2 Cellular decomposition

V práci [2] se autor zaměřuje na systematický průchod prostorem střídavými pohyby po sloupcích nahoru a dolů. Při tomto pohybu je agentem prostor dekomponován na nezávislé podprostory zvané buňky. Celý prostor je zde reprezentován grafem, kde buňka představuje vrchol grafu, ve kterém jsou hranou propojeny sousedící buňky. Prostor každé buňky je agentem procházen nezávisle na ostatních a hlavním problémem je nalezení nejkratší cesty v tomto grafu sousedů. Na počátku vnímá agent prostor jako jednu buňku a až při nálezu překážky začne prostor dělit, tak aby současná buňka měla dva nové sousedy, reprezentující prostor nad a pod překážkou. Agent zde průzkum končí, pokud má prozkoumané všechny známé vrcholy grafu a zároveň narazil na pravý okraj mapy.



Obrázek 3.3: Ukázka způsobu průchodu agenta prostorem. Převzato z [2].



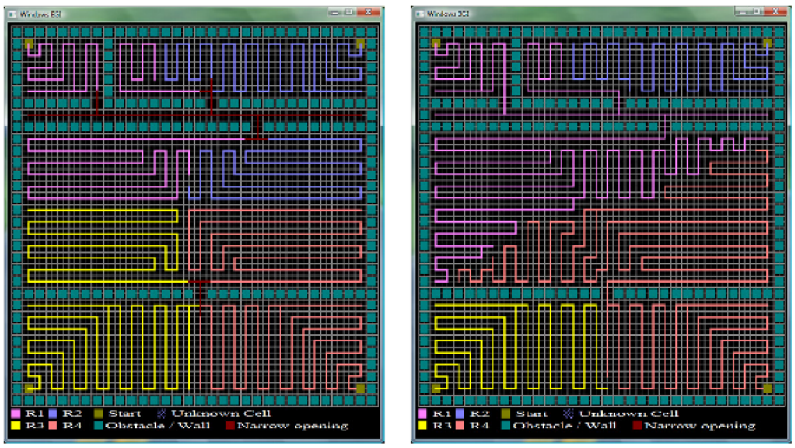
Obrázek 3.4: Prozkoumaný prostor tvořící graf sousedů o čtyřech vrcholech. Částečně převzato z [2].

Ve výše zmíněné práci se uvažuje pouze statický prostor s jediným agentem, ale při možnosti komunikace lze uvedené principy využit i v dynamickém multiagentním systému. Při nepřetržitém průchodu by vrcholy grafu byly agentům přidělovány vhodnou metrikou, např. nejbližší vzdálenost a při nalezení nekonzistence vjemů s bází znalostí by byl graf aktualizován sloučením či rozdělením vrcholů.

3.1.3 Spanning tree

Práce [10] při které není potřebná žádná znalost terénu má několik variací, ale společným základem je procházeni skupiny agentů prostorem, přičemž si každý udržuje strukturu lineárního seznamu skládající se z předchozích navštívených bodů, zde nazývaný jako "spanning tree". Úvodní průzkum probíhá pohybem agenta proti směru hodinových ručiček, tak

že neustále *přímo* obchází doposud vytvořený strom. Konec nastává pokud neexistuje na agentově pozici neprozkoumaná buňka. Následně vzniká dle počtu agentů X těchto "spanning tree", které určují agentovu nejdelší možnou Hamiltonovskou cestu. Algoritmus se může pro optimalizaci pokusit tyto cesty přerozdělit, či využít heuristik při průzkumu, aby byly cesty mezi agenty vyvážené.



Obrázek 3.5: Základní (napravo) a modifikovaná (nalevo) verze algoritmu. Převzato z [10].

V nejjednodušší verzi nedochází k přerozdělení terénu či jiným optimalizacím, což vede na nerovnoměrnost prozkoumávaných oblastí, kdy jeden agent může být zablokován a zbytečně čekat a další agent zkoumá zbylý nepřístupný prostor. Modifikace zde spočívá ve schopnosti agenta po uzavření svého prostoru sdílet již prozkoumaný prostor s dalším agentem a přesunout se takto do další oblasti. Tento přístup přináší další výhodu, při možném výpadku agenta jsou totiž schopni ostatní pokračovat v jeho práci.

3.1.4 Market economy

Práce [18] se při průzkumu inspiruje principy trhu. Agenti při průchodu prostředím generují seznamy úkolů k prozkoumání pomocí definovaných strategií, které budou popsány dále. Tyto úkoly značí pozici k prozkoumání na mapě. Tento seznam je seřazený tak, aby celková délka cesty byla co nejnižší. Následně se agent pokusí "prodat" své úkoly na způsob aukce. Ostatní agenti na tyto úkoly dávají nabídky zohledňující jejich potenciální profit. Úkol je prodán nejvyšší nabídce, pokud překročí určenou minimální cenu. Po ukončení všech aukcí agenti započnou cestu k nejbližšímu úkolovému bodu, po jehož dosažení vygeneruje další úkoly. Počet nově generovaných úkolů zohledňuje současný počet úkolů v systému, při velkém počtu by mohlo dojít ke zpomalení systému kvůli náročnosti výpočtu aukcí a cest. Následně se proces cesty k nejbližšímu bodu a aukce ostatních opakují.

Bod k prozkoumání je vlastněným agentem zaveden do aukce s cenou rovnou agentovu zisku z úkolu, pokud by jej agent zakomponoval do své cesty. Zisk je pro agenta funkce zohledňující počet neprozkoumaných buněk v okolí cíle a vzdálenost agenta od tohoto cíle. Nabídky aukcí B_i jsou spočteny každým agentem na potenciálním zisku při přidání do své cesty následovně:

$$B_i = P_r + \alpha * (v_i - P_r) \tag{3.4}$$

v_i je potenciální zisk agenta, P_r je prodejcem navržená cena a α je koeficient v intervalu 0–1, v práci je použito $\alpha = 0.9$. Toto číslo udává kompromis v poměru prodaných k zachovaným úkolům. Pokud nakupující agent očekává větší zisk, než by získal prodejce, bude platit $B_i > P_r$ a agent získá úkol v případě, že jiný agent jeho nabídku nepřevýší. V opačném případě platí pro všechny agenty $B_i \leq P_r$ a tedy prodávající agent si úkol ponechá, jelikož jej dokáže splnit nejefektivněji.

Úkoly zde vytváří komoditu trhu a jsou základní jednotkou při vytváření agentových cest, přičemž při jejich tvorbě agenty lze využit různých heuristik:

- *náhodně* - vybere se náhodný bod na mapě. Tento výběr je případně opakován do doby, dokud místo není obklopeno dostatkem míst neprozkoumaných a má jej tedy cenu zkoumat.
- *chamtivý průzkum* - cílový bod je vybírán z nejbližších neprozkoumaných oblastí.
- *čtyřstrom* - neprozkoumané buňky jsou mapovány do struktury čtyřstromu. Prostor je rozdělen na 4 uzly stromové struktury, pokud je jeho značná část neprozkoumána. Tento proces probíhá rekurzivně a končí, pokud je velikost prostoru menší než dohledová vzdálenost robota. Cílové body jsou středy listových oblastí těchto stromů.

Z výsledků dané práce se jeví nejefektivnější využít strategie *náhodného výběru* a *čtyřstrom*.

3.1.5 Brick&Mortar

V následujícím článku je [4] k průzkumu využito algoritmu, který označuje buňky do jednoho z těchto stavů:

- *zeď* - agent danou buňkou nemůže projít
- *neprozkoumáno* - daná buňka nebyla navštívena nebo je její stav neznámý
- *prozkoumáno* - buňka byla agentem již prokoumána a není potřebná k průchodu. Z pohledu chování agenta je ekvivalentní s typem *zeď*
- *navštíveno* - buňka byla agentem alespoň jednou navštívena, ale bude znovu využita k průchodu do jiných *neprozkoumaných* prostorů

Principiálně fungování spočívá v postupném zmenšování neprozkoumaného prostoru buňkami typu *prozkoumáno*, tento typ je nastaven vždy, pokud daná buňka nespojuje dvě oblasti, ve kterých je očekávaný budoucí průchod. Algoritmus je ukončen, pokud se v agentově okolí nenachází neprozkoumané oblasti. Při využití více agentů jsou pole rovněž označována ID příslušného agenta, aby nedošlo k situaci uvěznění jiným agentem, a tudíž byla spolupráce efektivnější. Dané řešení zaručuje prozkoumání celé mapy, ale pouze ve statickém prostředí, v měnícím se prostředí je jeho použití nevhodné.

3.1.6 Frontier-based exploration

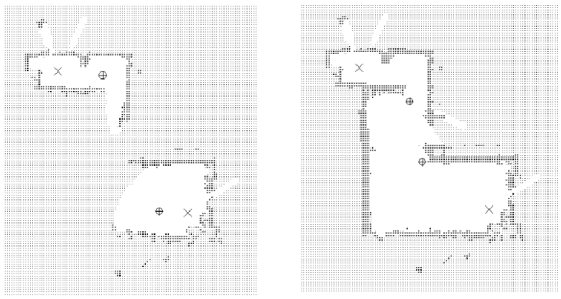
Senzory reálných robotů většinou mají větší dosah omezený převážně překážkami, tohoto lze využít k optimalizaci průzkumu terénu, na nějž se zaměřuje práce [15].

Robot se zde řídí mapou hranic svého dohledu, tedy přechodů mezi známým a neprozkoumaným prostředím. Tyto přechody se následně dle vlastní vzdálenosti shlukují a jsou

přidány do seznamů k prozkoumáni pouze pokud přesáhne počet výskytů určenou hranici (většinou velikost robota). Výběr sektoru k průzkumu může být dle vzdálenosti od robota, nebo podle velikosti shluku. Větší shluky pravděpodobně skrývají větší neprozkoumanou oblast a maximalizují tedy v minimálním čase množství získaných informací.

V případě průzkumu více roboty je společnou komunikací vytvářena globální mapa, ve které je agentům přidělována centrálním agentem oblast k průzkumu. Alternativně si agenti mohou oblasti vybírat nezávisle na sobě, což s sebou nese výhodu decentralizovaného přístupu odolného vůči výpadkům agentů s nevýhodou pomalejšího průzkumu, jelikož více agentů může zvolit stejnou oblast k průzkumu.

Algoritmus předpokládá jednoduchý průchod terénem se statickým prostředím. Pro dynamické prostředí by šla oblast modifikovat časovým razítkem, po jehož vypršení by byla oblast anulovaná a bylo ji nutné znovu prozkoumat.



Obrázek 3.6: ukázka postupné tvorby mapy. Převzato z [15].

3.2 Konkurenční týmy

Pořadatel po každém ročníku soutěže přichází s publikací komentující výsledky daného ročníku soutěže [1]. Soutěžícím týmům je při této příležitosti věnována kapitola, ve které vysvětlují své snahy, postupy a záměry. Nabízí se tedy nahlédnout a případně se inspirovat jejich postupy při průzkumu terénu.

FIT BUT

Každý agent našeho týmu si zde buduje mapu všech známých objektů na mapě spolu s jejích časovým razítkem. Pokud se dva agenti poznají, snaží se tyto mapy spojit s ohledem na jejich časová razítka. Samotný průzkum probíhá náhodným výběrem hraniční oblasti mapy nebo přezkoumáním dlouho nenavštívené oblasti. Pokud se dva již synchronizovaní agenti potkají a nesouhlasí jejich vzájemná pozice od očekávaného vektoru vzdálenosti uloženém ve společné bázi znalosti, muselo dojít k překročení hranice mapy a lze z rozdílu těchto vektorů spočítat očekávaný rozměr mapy.

GOAL-DTU

Tým Dánské technické univerzity si kvůli proměnlivému prostředí ukládá do společné báze znalosti pouze pozice neměnných objektů, tedy vydavače bloků, goal zóny a task board, přičemž problém nekonečnosti mapy je řešen obdobně jak týmem FIT BUT. Samotní agenti si

ve svých relativních mapách ukládají časy navštívení buněk a díky pseudonáhodné heuristice vybírají pro průzkum spíše déle neprozkoumaná místa. Díky nedostatečnému ukládání do společné báze dat toto může vést na opakovaný průzkum stejného prostoru více agenty a má tedy průzkum v pozdějších fázích spíše zanedbatelný význam. Samotné plánovaní cest je prováděno A* algoritmem a je spíše reaktivního charakteru.

MLFC

Tým Liverpoolské a Manchesterské univerzity spoléhal z předchozích ročníků na známou velikost mapy, tudíž nově zavedl roli "kartografa". Kartografem se stane dvojice agentů na počátku soutěže a má za cíl se pohybovat opačnými směry. Při následném střetnutí jsou pak schopni zjistit velikost mapy. Průzkum terénu probíhá náhodným výběrem směru, ve kterém agent pokračuje, dokud nenarazí na překážku. Do společné báze znalostí se obdobně ukládají pouze statické objekty. Plánovaní cest taktéž vychází hlavně z vjemů agentů.

LTI-USP

Tento Brazilský tým k průzkumu terénu zvolil obdobnou taktiku jako tým MLFC. Mírnými změnami však je omezení maximální vzdálenosti agenta od počátku a 1% šance změny směru při průzkumu, aby bylo zamezeno nekonečnému průzkumu pouze jedním směrem.

JaCaMo Builders

Skupina JaCamo je taktéž z Brazílie, ta pro minimalizaci času stráveného průzkumem terénu využila výše popsaný spanning tree algoritmus modifikovaný o delší dohled agenta. Zkoumání terénu je taktéž zaměřeno hlavně na získání informací o statických elementech. Pozdější přezkoumávaní terénu tedy není příliš důležité a při plánováni tras pomocí A* je vycházeno primárně ze senzorů agenta. Pro zjištění velikosti mapy je použit obdobný postup protichůdných agentů popsaný výše u týmu MLFC.

Kapitola 4

Implementace

V této sekci je popsána existující infrastruktura platformy deSouches a její komunikace se serverem. Následně je popsán vývoj algoritmu pro průzkum terénu a nutné změny kódu pro jeho zakomponování do současného programu.

4.1 MAPC2021 – současná platforma

Jako základ pro letošní soutěž byla využita platforma z roku 2021 vytvořena čistě v jazyce Java s využitím knihovny JADE [11]. Tato knihovna slouží jako základ pro vývoj univerzálních multiagentních systémů. Poskytuje grafické rozhraní pro správu agentů a jednotné rozhraní pro distribuovanou multiagentní komunikaci mezi množinou agentních stanic, bez ohledu na platformu a využití agentů. Tato komunikace je standartně stavěna na jazyku FIPA-ACL, ale umožňuje i jiné formy komunikace, pro naše účely je s předpokladem běhu všech agentů na jediné stanici využita sdílená paměť mezi agenty, pomocí které si zasílají příkazy a sdílí vjemy.

Vstupní bod programu

Jako vstupní bod programu slouží třída *DeSouches* pojmenovaná jako celý program na počest generála bránícího Brno za třicetileté války před švédskou invazi. V této třídě jsou nejprve inicializování agenti v separátních vláknech s využitím lokálních konfiguračních souborů *json* formátu a informací poskytnutých soutěžním serverem. Tyto informace zahrnují obecné parametry jako počet agentů v kole soutěže, jejich dostupné role a dobu po kterou se čeká na příkazy od agentů pro daný krok. Ke každému agentovi je odeslán identifikátor, ke kterému se lokální entita agenta autentizuje a je po dobu soutěže nebo do výpadku provázán. Navíc jsou zde inicializována různá grafická rozhraní pro monitorování běhu agentů. Jedno z těchto oken sloužící ke sledování obecného stavu agentů je vytvořeno přímo knihovnou JADE. Takovéto okno je užitečné především pro jednoduchou kontrolu živosti a aktivity agentů. Navíc je pro naše potřeby vytvořeno okno s detailnějšími informacemi pro každého agenta. Toto okno zahrnuje zejména jejich vjemy, provedené akce a jejich představy globální mapy.

Agent a jeho řídící smyčka

Dle specifikací knihovny JADE byla vytvořena třída DSAgent vycházející z třídy *JADE.Agent*. Počet inicializovaných instancí je specifikován požadavky serveru na počet agentů v daném

kole a každý z těchto agentů je vytvořen ve vlastním vlákně komunikujících navzájem pomocí společného generála a sdílených referencí na zdroje. Instance agenta po prvotní inicializaci a autentizaci k serveru má jako hlavní úkol vykonávaní procedury *action*. Tato procedura se vykonává s každým krokem simulace a obsluhuje následující činnosti: Na počátku kola simulace je serverem zaslána zpráva obsahující vjemy agentů, úkoly a platné normy, což jsou omezení při jejichž porušení jsou agenti potrestání srážkou energie. Tyto vjemy jsou pomocí dodané knihovny "eismassim-X.X-jar-with-dependencies" přijaty a rozděleny mezi agenty, kteří si z nich nejprve vytvoří samostatnou mapu výhledu okolí, jelikož tyto data přijdou s relativními souřadnicemi vůči agentovi. Následně jsou tyto data z výhledu agenta transformována do mapy agenta pomocí známého vektoru souřadnic agenta od svého počátku. Tyto data jsou pomocí obdobného systému transformována do mapy skupiny. Pokud agent narazí na nového týmového agenta, pokusí se s ním synchronizovat a spojit mapy pro udržení více informací o prostředí a možnost komunikace se společnou soustavou souřadnic. Po počáteční fázi vjemů si agent vytvoří seznam úkolů s danými prioritami načež si nejdůležitější z nich vybere a začne vykonávat s využitím výše zmíněné knihovny zasíláním akcí agenta v *json* formátu na server. Tento postup se opakuje s každým krokem do konce simulace.

```
def agentmainLoop():
    # SENSING
    percepts = waitForPercepts()
    outlook = createOutlook(percepts)

    updateGroupMap(outlook)
    updateGui(outlook, self.getGroupMap())

    # LIVELOCK DETECTION
    checkActivity(self)

    # EXECUTION - PREPARE
    synchronizeGroups(self, outlook, allKnownAgentsList)
    createGoalsList(self)

    # GOAL EXECUTION
    goal = getHighestPriorityGoal(self)
    sendActionToServer(goal)
```

Výpis 4.1: pseudokód agentovi řídící smyčky

Volba úkolu

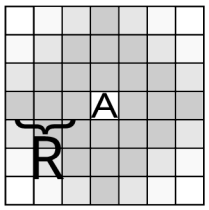
Pokud agent zrovna aktivně nevykonává úkol zadaný generálem vytvoří si v tomtéž kroku simulace seznam úkolů, ze kterého následně vybírá. Seznam úkolů je tvořen ze všech tříd dědících z třídy *DSGoal*, načež je vybrán nejdůležitější. Mezi tyto úkoly patří například: průzkum, čištění překážek nebo odevzdání úkolu. Vzhledem k nepředvídatelnosti systému je navíc implementován systém pro detekci zaseknutí, který například v případě zablokování překážkami umožní opustit od současného úkolu a započít únikovou akci. Po výběru prioritního úkolu je na dalších několik kroků simulace zahájena jeho činnost a příslušné informace jsou průběžně aktualizovány v grafickém rozhraní.

Průzkum terénu a synchronizace agentů

Zkoumání prostoru je v kostře platformy definováno v třídě *DSGoalExplore* a probíhá pomocí výběru náhodného bodu ve fixní vzdálenosti tří. Kvůli neoptimálnosti algoritmu mu není přikládána velká váha a je tedy označován nejnižší prioritou 1. Při průzkumu terénu o sobě agenti zpočátku nemají informace a musí se vzájemně identifikovat, aby nevědomě neprozkoumávali stejný prostor, ale mohli spolu sdílet vjemy a snadněji spolupracovat. Serverem jsou na počátku kroku simulace získány v rámci vjemů pozice všech agentů, bez jejich bližší identifikace. Agenti poté v rámci třídy *DSSynchronize* ukládají vektory viděných agentů do sdílené paměti. Agent je následně schopen detekovat, zda je v rámci kroku simulace poslední z ukládajících agentů díky serverem poskytnuté informaci o počtu agentů v rámci kola soutěže. Pokud agent zjistí, že je poslední, započne synchronizaci. Ta se skládá z porovnávání viděných vektorů všech kombinací agentů. Pokud je nalezena jednoznačná shoda vjemů tito agenti vytvoří skupinu. Velitelem skupiny je vždy agent s nižším číselným identifikátorem a do jeho mapy jsou přidávaný nové informace. Jelikož agenti pracují s relativními souřadnicemi, musí být před sloučením jejich souřadné systémy vhodně upraveny, pomocí známé pozice agenta od počátku své mapy a pozice slučovaného agenta. Tento proces je opakován dokud nevznikne jediná mapa s agentem A1 jako velitelem obsahující všechny agenty.

4.2 Návrh a implementace algoritmu

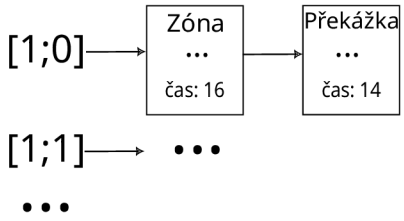
Pro implementaci jsem plánoval zvolit dříve zmíněný systém aukcí 3.1.4, jehož hlavní výhodou je snadná možnost dynamického přerozdělování úkolů v měnícím se prostředí. Avšak agenti se musí aktivně účastnit aukce, při jejíž výhře zablokují ostatním agentům daný úkol, což se ukázalo jako nevýhoda. Univerzální systém vytváření úkolů z tříd *DSGoal* totiž umožnuje agentovi vyhrát aukci, načež daný průzkum nevykoná, jelikož dostane zároveň prioritnější úkol. Jiný agent zároveň nemusí mít nic užitečnějšího na práci a daný průzkum by pro něj byl optimální činností. Řešením by byla eskalace priorit daného úkolu při výhře optimální aukce, ale potom by jiné důležitější úkoly mohli být ignorovány, proto jsem od tohoto nápadu upustil. Tato nemožnost sledování aktivně zkoumajících agentů vedla na více reaktivní přistup pomocí algoritmu na bázi optimalizací mravenčích kolonií.



Obrázek 4.1: ukázka vypuštění feromonů vzhledem k dohledové vzdálenosti R

Agent při průchodu terénem ve své dohledové vzdálenosti vypouští předem určené množství feromonu, jenž je realizováno pomocí neustálé aktualizace buněk v dohledové vzdálenosti. Tyto buňky si následně množství feromonu pamatují díky pomocné proměnné. K takovémuto kroku bylo nutné modifikovat koncept ukládaní dat agentů, kteří si doposud ukládali pouze relevantní informace poskytnuté z vjemů serveru, mezi něž patří například zóny a překážky, ale nikoliv volné buňky. Tyto volné buňku bylo nutné implementovat jako

doplněk dostupných vjemů s ohledem na dodané informace, jelikož například zóna se může vyskytovat na volném prostoru, ale i nad překážkou.



Obrázek 4.2: ukázka formátu uložení informací agentů v mapě

S tímto problémem se pojí ukládání informace pro každou souřadnici v prostoru v lineárně vázaném seznamu a udržení konzistentních informací mezi prvky těchto seznamů. Při přidávaní nových informaci je třeba kontrolovat platnost volných buněk. Každá z buněk seznamu navíc může mít jiné časové razítko objevení, a tedy i jiné množství feromonu.

4.2.1 Implementace feromonů

Vztah mezi množstvím feromonu a dobou od průzkumu buňky tvoří nepřímou úměru. Nejnovější buňky dosahují nejvyšších hladin látky, a tudíž by měla být minimalizována snaha o jejich prozkoumání. Aby bylo tohoto chováni dosaženo je třeba implementovat procedury simulující přirozené chování těchto feromonů. Jedním z nich je *vypařování*, které zaručuje snížení množství feromonu s pokračující dobou simulace a zajišťujíc tak stárnutí informace. Druhou z těchto vlastností je *difuze*, která pomocí průměrování okolních hodnot zaručuje zohlednění informace z širšího okolí při rozhodování o dalším kroku pro informovanější rozhodnutí oproti výběru ze samotné neprozkoumané buňky, která může být již z celého okolí prozkoumána, či obklopena podavači.

Kvůli nejistotě a nezaručitelnosti vykonání procedury *DSGoalExplore* bylo nutné volat procedury *vypařování* a *difuze* přímo z povinného bloku *action*, který má zaručenou vykonání každým aktivním agentem v každém kroku simulace. Ideálně by pro účely přepočítáváni mapy feromonů byl vyhrazen jediný agent skupiny sdílející tuto mapu, ale kvůli neustálým změnám, spojování těchto skupin a celkové volatilitě informací se ukázal tento přístup nevhodný, jelikož není zaručena stálost členů skupiny ani při procesu tohoto přepočítávání. Proto spolu agenti sdílí singleton třidu, kterou se v každém kroku pokusí všichni existující agenti zaktualizovat. Pomocí ukládání kroku simulace poslední aktualizace pro každou existující skupinu a synchronizačních mechanismů, je ale zaručeno, že mapa bude upravena právě jednou pro každou aktivní skupinu.

Při procesu aktualizace je nejprve vytvořena virtuální mapa známého prostoru. Jelikož agent nezná absolutní velikosti mapy, počítá s doposud známými extrémy souřadnic známé mapy, z jejichž kombinací zjistí souřadnice levého horního a pravého dolního rohu, které jsou využity pro odhadovanou velikost mapy. Tato mapa je následně doplněna o všechny neznámě buňky s nulovou hladinou feromonu, pro vytvoření snahy o jejich průzkum. Mapa je pak systematicky procházena a pro každou buňku je spočtena difuze z okolí, načež je celá mapa aktualizována s využitím fixního koeficientu odpaření α. Pro neexistující buňky virtuální mapy jsou informace pro současný krok simulace zahozeny a pro další krok jsou počítány nanovo. Při výskytu více buněk pro jednu souřadnici je počítáno s nejnovější informací načež při aktualizaci je informace pozměněna pro všechny buňky dané souřadnice.

Původní myšlenkou implementace difuze bylo šíření feromonu do vzdálenosti omezené pouze vyprcháním se vzdáleností zaručeného pomocí vhodného koeficientu, tímto způsobem by bylo zaručeno nejvhodnější rozhodnutí pro zkoumání díky zohlednění informací z nejširšího okolí. Kvůli časovým omezením kroku simulace a v nejhorším případě exponenciální složitosti tohoto řešení však není možné takovouto mapu v daném čase vytvořit a bylo nutné hledat jiné řešení.

```
for cell in virtualMapCells():
    propagate(cell,cell.propagatePheromone)

def propagate(cell,amount):
    cell.pheromone += amount
    if amount is not 0:
        for p_cell in getNeighbours(cell):
            propagete(p_cell,reducePheromone(amount))
```

Výpis 4.2: pseudokód prototypu algoritmu difuze

Takovýmto řešení pracujícím v lineárním čase se jeví zohledňovat pouze fixní okolí. Po experimentech se pro difuzi ukázalo okolí do manhattanské vzdálenosti 2 jako dostačující. Výsledná stopa je následně vypočítána jako průměr z tohoto okolí.

```
for cell in virtualMapCells():
        amount = getNeighbours(cell).sumPheromones()
        cell.pheromone += reducePheromone(amount)
```

Výpis 4.3: pseudokód použitého algoritmu difuze

Matematicky by šlo množství feromonu buňky O_i vyjádřit následovně:

$$O_i = \alpha * average(f(i, 2)) \tag{4.1}$$

Kde funkce $f(i, 2)$ slouží k získání feromonu všech sousedů buňky i do vzdálenosti 2.

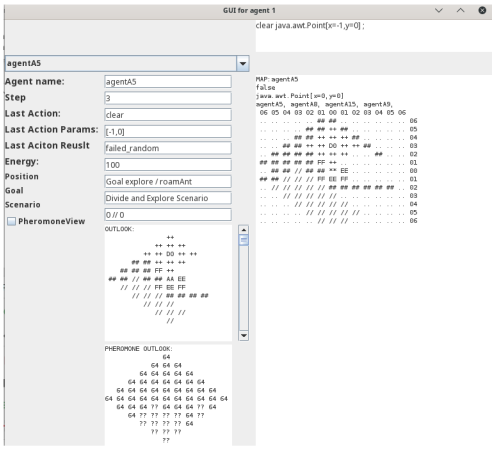
4.2.2 Průzkum agenty s využitím feromonů

Původní postup náhodného postupu při vykonávání úkolu *DSGoalExplore* byl pozměněn na přihlížení k feromonovým stopám sousedních buněk. Pro účely tohoto náhledu jsou vybírány buňky na hranici agentovi viditelnosti, tato vzdálenost byla zvolena jako kompromis mezi příliš vzdálenými body, které by potenciálně měli podobně nízkou hladinu pachové stopy a čistě reaktivním přístupem dohledu pouze na další agentův krok, jelikož tyto body jsou z agentova pohledu plně prozkoumány a je na ně minimálně propagována stopa neprozkoumaného okolí a mají tudíž z agentova pohledu minimální množství získaných informací. Výhodou této vzdálenosti je jistota existence těchto bodů v agentově mapě, což pro neprozkoumané území za hranicí agentovi virtuální mapy neplatí, což se projevuje zejména na počátku simulace, kdy agent není součástí vetší skupiny a pohybuje se po okrajích známé mapy. Pro průzkum je z množiny bodů v dané vzdálenosti vybrán ten s nejnižší feromonovou stopou. Může nastat situace taková, že cesta na daný bod je nedostupná nebo se stala nevhodná, například kvůli zablokování agenty, a tudíž by se daná cesta potenciálně prodloužila na větší vzdálenost přes již prozkoumaný terén. Pro tyto případy byly do množiny bodů k průzkumu přidány i další body ve stejné vzdálenosti. Tato množina je následně seřazena vzestupně vzhledem k pachové stopě a v daném pořadí jsou i testovány na existenci cesty.

Pokud cesta k žádnému z těchto bodů není možná žádný plán cesty není vytvořen a agent vykoná jinou vhodnou činnost. Při průzkumu je navíc zvýšená snaha o změnu role, pokud je *role area* v blízkosti, jelikož výchozí role je pro průzkum nevýhodná, role k přidělení je vybírána generálem z následujících možností: digger, worker a explorer. Podle těchto rolí se následně odvíjí využitý algoritmus průzkumu. Role explorer má zvýšený dohled, ale zato nemůže vykonávat clear akce, proto je využíván algoritmus A*, který se v této modifikaci vyhýbá překážkám, pokud taková cesta bez překážek není možná algoritmus selže a zkouší cesty na jiné pozice. Role explorer taktéž disponuje možností pohybu o více pozic v jednom kole. Tato role však nebyla na kostře platformy z minulého roku zohledněna. Ostatní role disponují pouze schopností pohybu jednoho kroku za kolo a bylo tedy nutné pozměnit jak akci *DSMove* pro ukládání vektoru jednotlivých pohybů vzhledem k využívanému rozhraní knihovny serveru, ale také bylo nutné pozměnit samotné plánovací algoritmy, aby této možnosti využívaly. Ostatní role využívají algoritmu plánování "přímé cesty", která plánuje nejkratší možnou cestu skrz překážky za pomocí clear akcí a je zastavena pouze přítomností nepřekročitelného agenta na dané pozici. Tento algoritmus je výhodný, jelikož buduje snazší cestu pro průchody dalších agentů. Nejvhodnější pro tento algoritmus přímé cesty je role digger se 100% úspěšností clear akcí narozdíl od pravděpodobnosti 30% pro výchozí roli.

4.2.3 Grafické uživatelské rozhraní

Grafické rozhraní zde hraje spíše druhořadou roli k ladění chyb, jelikož server již poskytuje nástroje k monitorováni průběhu soutěže. Formulářové okno postavené na platformě swing, zůstává od minulého ročníku spíše nezměněné, ale bylo zpřehledněno listování agentů a po vzoru současného ASCII výpisu byla přidána možnost sledovat feromonovou mapu jak jednotlivých agentů, tak mapy skupinovou. Hodnoty jsou zde zapsány v hexadecimálním formátu, abychom se na buňku po vzoru současné mapy vmístili do dvouznakového zápisu. Agenti jsou na této mapě označeni znaky "**".



Obrázek 4.3: ukázka uživatelského rozhraní aplikace DeSouches

4.2.4 Synchronizace agentů do skupin

Ačkoliv kostra již obsahovala nástroje pro spojování do skupin, její nedostatky se projevovali zejména při větším množství agentů, kde většinou existuje více páru agentů se stejnými

vektory a agent tak není schopen je odlišit tak je proces synchronizace do skupin oddálen. Pro porovnání agentů proto byl proto nově využit vektor všech vjemů, jež mají agenti ve společné dohledové vzdálenosti. Tato forma synchronizace taktéž není 100%, ale je významně nižší pravděpodobnost že více párů agentů bude sdílet stejný vjemový prostor. Obdobného mechanismu bylo využito pro zjištění velikosti mapy, kdy agent jednoznačně identifikuje "neznámého agenta", ale zároveň zjistí, že se jedná o agenta ve stejné skupině mimo jeho údajnou dohledovou vzdálenost, v tomto případě muselo dojít k překročení mapy a ze vzdálenosti těchto dvou agentů z báze dat a výhledového vektoru lze následně tato velikost spočítat. Tato operace však nezaručí zjištění obou dimenzí zároveň a postup může být nutné opakovat pro zbývající osu. Zároveň je možné, že agent překročí mapu dva a více krát, než narazí na agenta k synchronizaci. V tomto případě by bylo možné brát v potaz nejnižší možnou velikost mapy, kterou agenti zjistí, avšak kvůli nepravděpodobnosti takové události na ni nebyl brán zřetel.

Kapitola 5

Vyhodnocení

Pro účely vyhodnocení kvality a testování zvolených koeficientů algoritmu byla vytvořena třída *AntMapStatistics*. Metody této třídy jsou volány po dokončení činnosti *antMapUpdateSingleton*, tedy nad aktuální verzí mapy pro dané kolo. Třída *AntMapStatistics* se skládá z metod, které by šli rozdělit na 2 podtřídy: skupinové a obecné. *Skupinové* jsou kalkulovány pro každou skupinu v daném kroku samostatně a *celkové* se zabývají primárně agregovanými statistikami a jsou tedy počítány pouze jednou v daném kole. Požadovaných charakteristik výpočtu je zde dosaženo pomocí počítadel ukládajících krok poslední návštěvy a synchronizačních mechanismů, takže je zamezeno synchronnímu vykonání více agenty zároveň a pouze první z dané množiny statistiky počítá. Statistiky za běhu programu průběžně ukládány pomocí integrované logovací knihovny do různých csv souborů v adresáři *logs* pro následnou analýzu. Společným rysem těchto souborů je existence popisující hlavičky a každý řádek souboru se skládá z kroku simulace, názvu zapisující agentní skupiny doplněné v následujících sloupcích o samotnou informaci.

Pro testování a porovnání byly sledovány parametry jako počet agenty objevených pozic, jejich údajné velikosti mapy, rychlosti konvergencí k singulární skupině a další parametry sledující charakteristiky hlídkování prozkoumaného terénu. Inspirací pro sledované vlastnosti algoritmu byla práce [3]. K charakteristikám sledovaných pro jednotlivé buňky patří:

- Instantaneous Node Idleness (INI): Počet kroků mezi znovunavštívením buňky. Kritérium je počítáno pro každou buňku zvlášť
- Instantaneous Graph Idleness (IGI): Průměr INI z celého systému v daném čase
- Instantaneous Worst Idleness (IWI): Nejvyšší možný čas mezi opětovným navštívením buňky v daném systému

Ačkoliv implementace synchronizace agentů do skupin byla úspěšná a agenti většinou správně určují velikosti mapy, kvůli časové náročnosti procesu se může naskytnout chyba výpočtu. K chybě dochází zejména při vysokém počtu agentů, kvůli nedostatečnému času kroku simulace, tudíž se některý z agentů může od počátku výpočtu přesunout a nastane tak desynchronizace projevující se hlavně špatným výpočtem velikosti mapy, z tohoto důvodu nebylo s určenou velikostí mapy pracováno a agenti mapu překračují a považují ji za nekonečnou, i tak lze tento chod považovat za validní a jejich charakteristiky porovnávat. Toto chování navíc vysoce snižuje charakteristiky hlídkování algoritmu a soustředí se tak spíše na "nové" oblasti.

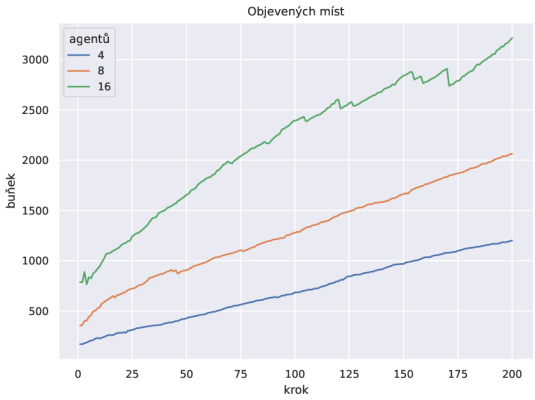
K analýze vytvářených csv souborů byl navíc v adresáři "statAnalyzer" vytvořen python skript, který využívá ke své činnosti primárně knihoven matplotlib, numpy a pandas. Pri-

márním účelem bylo vytvořit přehledné a různě agregované zobrazení velkého množství dat. Výstup programu je tvořen různými grafy formátu pdf uložených v podadresáři "figures".

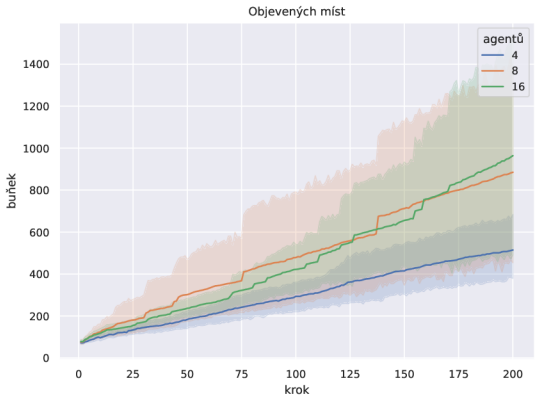
Konfigurace parametrů serveru jako počet agentů nebo nastavení prostředí je prováděno změnou konfiguračních json souborů ve složce *massim_2022/server/conf*. Obvykle jsou tyto simulace vedeny do kroku *750*, avšak pro účely testování bylo zvoleno *200* jako vhodný kompromis, nad který by už průzkum neměl mít pro agenta velký význam. Průzkum by totiž měl být hlavní prioritou spíše v počáteční fázi simulace a následně by mělo hrát prim spíše plnění úkolů.

Testovací data byla získána průměrem ze 3 běhů s různými počátečními podmínkami serveru pro zmírnění vlivu náhodných akcí na měření. Pokud není zmíněno jinak jsou využity výchozí hodnoty $\alpha = 0.9$ s počtem 16 agentů dle 4.1.

Škálovatelnost systému



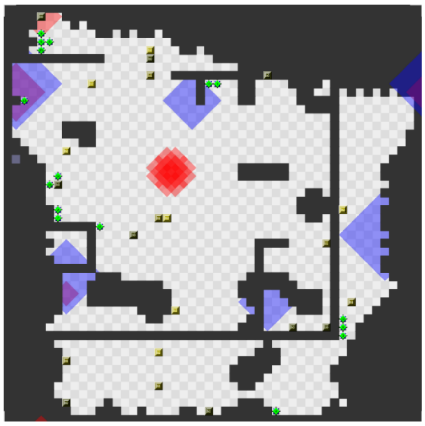
Obrázek 5.1: agregované zobrazení



Obrázek 5.2: detailní zobrazení znázorňující odchylky skupin

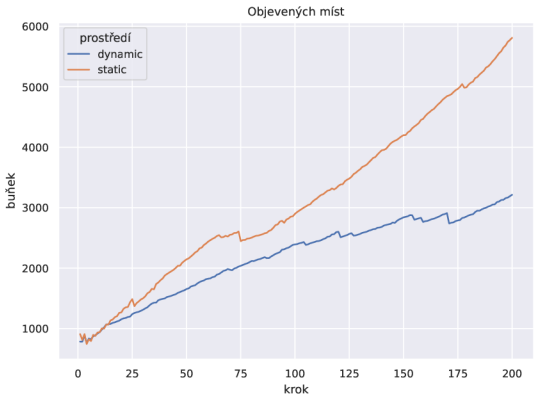
Soutěž se klasicky skládá z několika kol, každé s jiným počtem agentů, proto bylo důležité sledovat jak se systém chová s jejich měnícím počtem. Dle očekávání lze vidět zhruba lineární závislost mezi počtem agentů a prozkoumaným prostorem. Od lineárního trendu se však vyskytuje odchylka obzvláště zde viditelná mezi 4 a 16 agenty. Očekávali bychom čtyřnásobné množství prozkoumaného terénu ve prospěch 16 agentů, avšak prakticky je prozkoumaného terénu méně. Částečným důvodem tohoto jevu jsou překážky mapy a nutnost jejich odstranění. Pravděpodobnost úspěšností clear akce, která je k tomuto odstraňování využita, pro výchozí roli je pouze 0.3 a zřejmě se její vliv nejvíce projeví s vyšším množstvím agentů. Vliv zde však mohou mít i náhodné clear akce od serveru, které mají schopnost agenta vyřadit na několik kol z provozu, přičemž s rostoucím počtem agentů roste i pravděpodobnost zásahu touto akcí. Za zmínku stojí klesající segmenty grafu, které mají více příčin, v případě větších skoků na grafu 5.1 je vliv přičiněn spojování různých skupin a tudíž jsou odstraňovány duplicity. Menší lokální poklesy jsou zapříčiněny vlivem paralelnosti programu a s ní spojenou volatilitou dat, tedy přepisováním a odstraňováním buněk jinými agenty v rámci skupiny.

Vliv prostředí

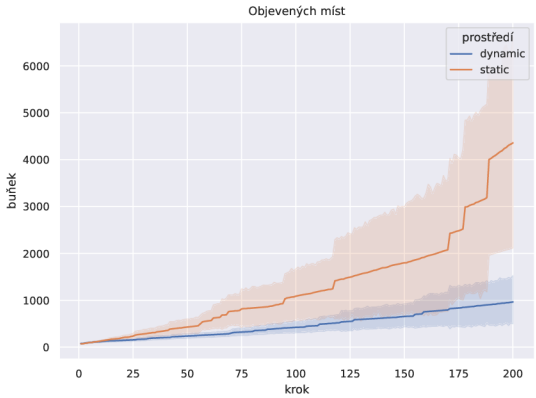


Obrázek 5.3: statické prostředí

V dalším testu byl zkoumán vliv prostředí na průzkum. Zkoumáno bylo otevřené prostředí s malým množstvím překážek 5.3, dále nazýváno jako “statické”. Clear akce serveru zde totiž mají nižší pravděpodobnost na ovlivnění agenta, jelikož vzniklé překážky může snadno obejít. Pro porovnání bylo zvoleno prostředí s vyšší četností překážek 2.1, dále nazýváno jako “dynamické”, jelikož clear akce serveru zde mají vyšší šanci a vliv při zablokování agenta. V případě dynamičtějšího prostředí jsou agenti negativně ovlivněni vyšším množstvím překážek, které jej jednak zpomalují v průzkumu kvůli nutnosti zdlouhavého odstraňování, ale také s tím související těžší synchronizací s ostatní skupinami. Významnou výhodou statického prostředí je jeho otevřenost a výskyt zón pro změnu rolí blízko počátečních pozic, agenti při výběru role “explorer” mají totiž zvýšenou pohyblivost a dohled. Vlivem těchto faktoru došlo až k 80% nárůstu prozkoumaných oblastí ve prospěch otevřeného statického prostředí.



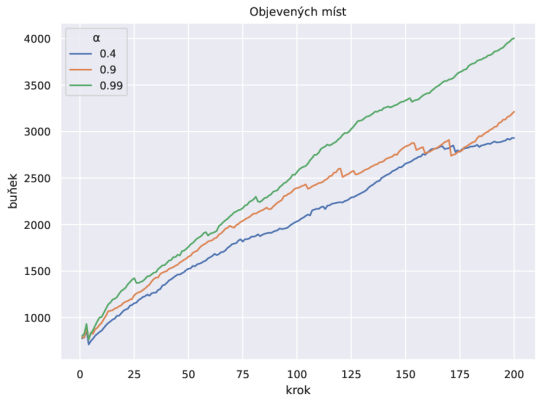
Obrázek 5.4: agregované zobrazení



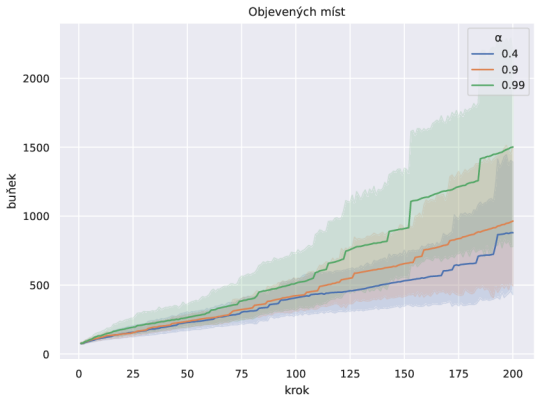
Obrázek 5.5: detailní zobrazení znázorňující odchylky skupin

Experimenty s koeficienty

Dalším pokus nastal ve změnách koeficientů vypařování pro Ant Colony Optimalization algoritmus (dále jen ACO) pro 16 agentů. Koeficient α v rozmezích 0 - 1 určuje jaký podíl účinné látky zůstane na konci kroku simulace zachován. Čím vyšší je toto číslo tím více má agent tendenci objevovat nové území oproti průzkumu známého prostoru. Pro $\alpha = 1$ by agent prozkoumával pouze nové území a za každou cenu se vyhýbal navštíveným, naproti tomu pro $\alpha = 0$ algoritmus degraduje na náhodný průzkum, jelikož agent ztratí veškerou informaci o předchozím průzkumu.

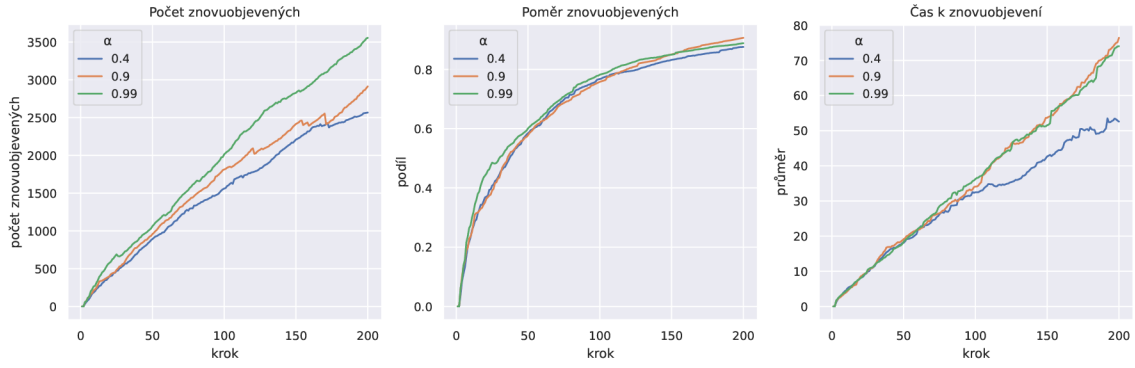


Obrázek 5.6: agregované zobrazení



Obrázek 5.7: detailní zobrazení znázorňující odchylky skupin

I přes to že agenti nepracují se známu velikostí mapy a jejich schopnost a sledování znovu navštívených buněk je tak degradováno, lze pozorovat určitý trend. Díky prostojům při clear akcích může látka dostatečně vyprchat a agent tak může uznat za vhodnější průzkum již dávno prozkoumaného prostoru oproti novému.



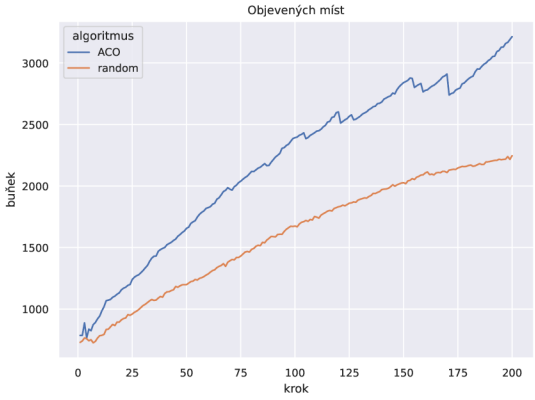
Obrázek 5.8: vliv koeficientu α na znovu navštěvování prozkoumaných buněk

Ve statistických údajích je zjišťováno především kolik pozic z celkového známého počtu bylo alespoň jednou znovu navštíveno a pokud ano jaký je průměrný čas, za který k němu došlo. Na grafu 5.8 lze vidět, že průměrný čas znovuobjevení úměrně klesá s hodnotou α. Zároveň lze pozorovat nejnižší počet objevených buněk pro $\alpha = 0.4$. Paradoxně $\alpha = 0.99$ zde vykazuje více znovuobjevených buněk než $\alpha = 0.9$. Tento jev bych přiložil náhodným

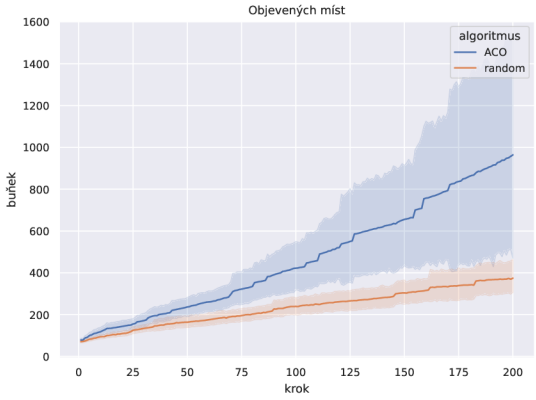
akcím serveru, které mohou mít vyšší vliv na výkonnost programu než zanedbatelná změna koeficientu odpařování.

Porovnaní s původní platformou

Dále se nabízí porovnání s původním řešením v kostře programu z předchozích let založeným na náhodném průzkumu. Pro algoritmus ACO bylo opět využito koeficientu $\alpha = 0.9$.

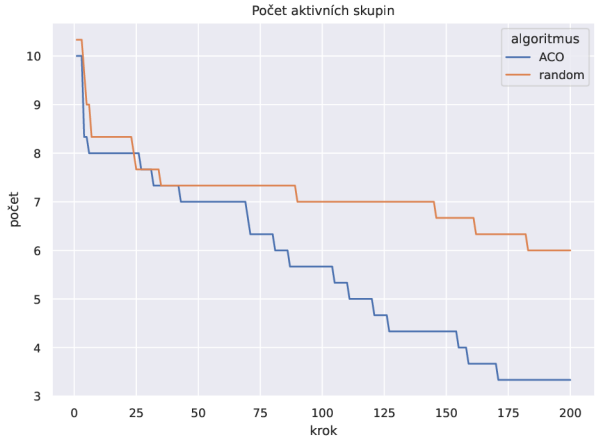


Obrázek 5.9: agregované zobrazení



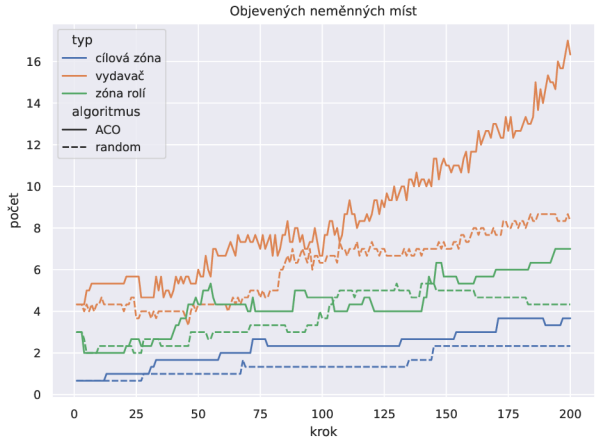
Obrázek 5.10: detailní zobrazení znázorňující odchylky skupin

Z grafu 5.9 lze pozorovat, že výsledky se liší zejména v pokročilé fázi simulace, jelikož při počátečních krocích je menší šance průzkumu známých prostředí náhodným algoritmem. S pokračující simulaci však dochází ke kombinatorické explozi a začne se projevovat nevhodnost neinformovaného řešení, kdy ke konci simulace došlo k více než 50% nárůstu prozkoumaných území ve prospěch ACO.



Obrázek 5.11: porovnání konvergencí skupin pro odlišná řešení průzkumu

Důležitým faktorem je také synchronizace a spojení skupin, kde je ideální dosáhnout co nejdříve jednotné skupiny obsahující všechny členy. Z grafu 5.11 je patrná lepší charakteristika a rychlejší konvergence nového řešení. Ačkoliv nebylo do konce simulace dosaženo jediné skupiny, byl jejich počet oproti původní metodě průzkumu zredukován na polovinu.



Obrázek 5.12: porovnání zjištěných neměnných bloků pro odlišná řešení průzkumu

Průzkum terénu je jistě důležitý, ale měl by v případě soutěže sloužit primárně jako podpora pro plnění úkolů zadaných serverem, proto je vhodné nahlédnout jak si ACO vedl při hledání míst důležitých pro tyto účely. Jmenovitě jde o hledání vydavačů bloků a cílových zón, které jsou navíc například oproti překážkám neměnné a tak tato získaná informace je užitečná až do konce kola soutěže. Z grafu 5.12 lze vidět jasný náskok ve prospěch ACO. Největším úspěchem je 16 vydavačů ACO oproti 8 u náhodného průzkumu. Opět podotknu existenci klesajících segmentů grafu, které jsou zapříčiněny z důvodu přehlednosti agregovaným zobrazením grafu sčítající znalosti všech skupin, tak v případě spojení je zjištěna duplicita a výsledný počet v grafu se tak sníží.

Kapitola 6

Závěr

Cílem této práce bylo prozkoumání různých řešení multiagentního průzkumu prostor a implementace vhodného řešení do kostry programu DeSouches pro minulé ročníky soutěže MAPC, které měli mírně pozměněná pravidla. Jako vhodná varianta se ukázala variace algoritmu optimalizace mravenčí kolonie, který narozdíl od jiných řešení nepotřebuje známou velikost mapy a pracuje s omezeným dohledem agenta. Notnou částí práce byla i celková změna struktury programu, aby odpovídal letošním požadavkům soutěže.

Implementovaný algoritmus vykazuje dobré výsledky, ačkoliv se vlivem náhodných faktoru soutěžního serveru budou konkrétní výsledky lišit z mých pozorování lze poukázat na některé kvantitativní údaje. Po 200 krocích simulace došlo ve prospěch mnou implementovanému algoritmem k o polovině více prozkoumaného území a sjednocení na poloviční množství skupin. Z pohledu faktorů důležitých pro další časti soutěže je vhodné zmínit i nalezení 2x více neměnných bloků důležitých k plnění úkolů zadaných serverem.

Mnou implementované synchronizační mechanismy, mají za následek rychlejší a spolehlivější spojovaní skupin agentů avšak, kvůli časové náročnosti celého procesu a možným race conditions se nebylo možné spolehnout na poskytnutý odhad velikosti mapy, ačkoliv byl většinu času přesný. Toto bych viděl jako největší potenciál pro zlepšení do budoucna, jelikož agenti by nezkoumali stejný prostor stále dokola s předpokladem že se jedná o nový prostor.

Osobním přínosem práce byla pro mě zlepšení se v orientaci v cizím kódu a s tím plynoucí nutnost spolupráce a zároveň zlepšení znalosti a odlaďování vícevláknových programů. Dalším hezkým bonusem bylo díky formátu soutěže získat vizualizace algoritmů probíraných v předmětu základy umělé inteligence.

Literatura

[1] Ahlbrecht, T., Dix, J., Fiekas, N. a Krausburg, T. The 15th Multi-Agent Programming Contest. In: *The Multi-Agent Programming Contest 2021*. Cham: Springer International Publishing, 2021, s. 3–20. ISBN 978-3-030-88549-6.

[2] Choset, H. Coverage of Known Spaces: The Boustrophedon Cellular Decomposition. *Auton. Robots*. Kluwer Academic Publishers. 2000, sv. 9, č. 3, s. 247–253. DOI: 10.1023/A:1008958800904. ISSN 0929-5593. Dostupné z: https://doi.org/10.1023/A:1008958800904.

[3] Chu, H. N., Glad, A., Simonin, O., Sempe, F., Drogoul, A. et al. Swarm Approaches for the Patrolling Problem, Information Propagation vs. Pheromone Evaporation. In: *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*. 2007, sv. 1, s. 442–449. DOI: 10.1109/ICTAI.2007.80. ISBN 978-0-7695-3015-4.

[4] Ferranti, E., Trigoni, N. a Levene, M. Brick amp; Mortar: an on-line multi-agent exploration algorithm. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, s. 761–767. DOI: 10.1109/ROBOT.2007.363078.

[5] Glad, A., Simonin, O., Buffet, O. a Charpillet, F. Theoretical Study of Ant-based Algorithms for Multi-Agent Patrolling. Červenec 2008. DOI: 10.3233/978-1-58603-891-5-626.

[6] Ing. František Zbořil Ph.D. doc. *Komunikace v MAS*. 2021. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FAGS-IT%2Flectures%2FAGS21_komunikace.pdf.

[7] Ing. František Zbořil Ph.D. doc. *Řešení konfliktů v MAS*. 2021. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FAGS-IT%2Flectures%2FAGS21_konflikty.pdf.

[8] Lesser, V. R. a Corkill, D. G. The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. *AI Magazine*. 1983, sv. 4, č. 3, s. 15. DOI: 10.1609/aimag.v4i3.401. Dostupné z: https://ojs.aaai.org/index.php/aimagazine/article/view/401.

[9] Prýmek, M. *Multiagentní systémy v praxi, implementace distribuovaného dohledového systému*. 2008. Disertační práce. MASARYKOVA UNIVERZITA, FAKULTA INFORMATIKY. Dostupné z: https://is.muni.cz/th/eqn7d/prymek_dp_final.pdf.

[10] Senthilkumar, K. a Bharadwaj, K. Multi-robot exploration and terrain coverage in an unknown environment. *Robotics and Autonomous Systems*. 2012, sv. 60, č. 1, s. 123–132. DOI: https://doi.org/10.1016/j.robot.2011.09.005. ISSN 0921-8890. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0921889011001783.

[11] team, J. *JAVA Agent DEvelopment Framework* [online]. Telecom italia, červen 2022 [cit. 2022-06-26]. Dostupné z: https://jade.tilab.com/.

[12] Technology, C. U. of. *The Multi-Agent Programming Contest* [online]. FIT VUT v Brně, červen 2005 [cit. 2022-05-03]. Dostupné z: https://multiagentcontest.org/.

[13] Wooldridge, M. *An Introduction to Multiagent Systems*. 2. vyd. Wiley, 2009. ISBN 978-0-470-51946-2.

[14] Wooldridge, M. a Jennings, N. R. Intelligent agents: theory and practice. *The Knowledge Engineering Review*. Cambridge University Press. 1995, sv. 10, č. 2, s. 115–152. DOI: 10.1017/S0269888900008122.

[15] Yamauchi, B. Frontier-Based Exploration Using Multiple Robots. In: *Proceedings of the Second International Conference on Autonomous Agents*. New York, NY, USA: Association for Computing Machinery, 1998, s. 47–53. DOI: 10.1145/280765.280773. ISBN 0897919831. Dostupné z: https://doi.org/10.1145/280765.280773.

[16] Zbořil, F. *Plánování a komunikace v multiagentních systémech*. 2004. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: http://www.fit.vutbr.cz/~zborilf/PhD/thesis.pdf.

[17] Zbořil, F. *Úvod do agentních a multiagentních systémů* [online]. FIT VUT v Brně, říjen 2006 [cit. 2022-03-22]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FAGS-IT%2Ftexts%2FAGS_opora_m1_ESF.pdf.

[18] Zlot, R. M., Stentz, A. T., Dias, M. B. a Thayer, S. Multi-Robot Exploration Controlled By A Market Economy. In: IEEE, ed. *Proceedings of (ICRA) International Conference on Robotics and Automation*. May 2002, sv. 3, s. 3016 – 3023.